Department of Computer Science
Computer Networks
Due: Sunday 8th September

Marking: 10% and up to 4 Bonus Marks

Your name:
TA Name:
Time Taken:
Estimated Time: 20 hours

This is the first of three programming projects for Computer Networks, and is worth a total of 10% of your final mark. Additionally 4 bonus marks may be awarded for this project.

You may work on this individually or as part of a group of no more than 3.

Be aware, code for this project may be reused if you wish for project 3. A modular approach to code organisation is always recommended.

Optional: Please include a rough estimate of how long it took you do the assignment so that we can calibrate the work being assigned for the course. (The estimated time is provided purely as a guideline.)

Important: Please read the instructions carefully. In particular:

- The programming language is C or C++
- You must provide clear instructions in a README file on how to compile and run your program, and what OS it was compiled on. If your program does not compile, please provide a detailed explanation of what that is. (2 marks)
- Code should be well commented. (2 marks)
- It is ok to use online sources for inspiration but you must write your own client and server, using good modularisation and appropriate variables. (For example, "someBool" is not an appropriate variable.)

Introduction

In this programming assignment you are asked to write a simple chat server, and accompanynig client, which can handle multiple simultaneous connections, following the specification below.

Programming Assignment

The goal of this assignment is to write a simple chat server using TCP/IP, which clients can connect to from any machine. You should plan to host the server on skel.ru.is, and run the clients from your local laptop.

As part of this project, you should also implement a simple access protection scheme called "Port Knocking" In port knocking a server listens on selected ports, and only allows connections that fit a predefined sequence.

For your server, you should pick 3 consecutive ports, that it will listen to, and only allow clients to connect that if they pick the correct sequence to connect to within a short time.

Note, this means that you should also check to see if anything else is listening on those ports. As your discovered in the last assignment, one way to do this is to try to connect() to a port.

For example: A server listens on ports 50001, 50002, 50003 for a client to try to connect to port 50003, 50001, and 50002 within 2 minutes, in that order. It only responds to the client on port 50002, and only if it has seen connections on the other two ports.

Note, to do this you will need to use non-blocking sockets, and the select() call. In addition to the online manual for socket and select, the following example may be useful:

`http://www.gnu.org/software/libc/manual/html_node/Server-Example.html`

Once it has sucessfully managed to connect to the server, the clients should setup and tear down a TCP/IP connection with the server, and use this to send text messages to the server, using the API described below.

The server is reponsible for sending the chat messages to the clients as appropriate. Once you have written your server, you should leave it running on skel (see bonus points).

Note: The nohup and disown commands can be used from a bash/zsh shell to run a command independently from the terminal, such as a server. If you do this be careful to clean up any stray processes.

API The server must support at least the following commands from the clients:

1. ID :: Provide a unique ID for the server.
   - The ID should be generated once, and hardcoded as a string in the server, by using the fortune command, *fortune -s* and adding the group's initials and a timestamp at the end.
2. CONNECT <NAME> :: Join chat on server with <Name>
3. LEAVE :: Disconnect from server
4. WHO :: List users on server
5. MSG <USERNAME> :: Send message only to the user specified
6. MSG ALL :: Send message to everybody
7. CHANGE ID :: Change the ID on the server.
   - This command should run fortune -s (use system()), and create a new ID, again with a timestamp and the group's initials at the end.

Bonus Points You may optionally add code to your server to scan for other servers on skel, and attempt to capture their ID's. You will be awarded 1 bonus point for each unique ID that you capture up to a maximum of 3.

You will be awarded one bonus point if at least two groups capture a unique ID from your server and reports it for bonus points. To receive bonus points for this the ID each group reports must be unique. This means to receive bonus points you must use the CHANGE ID command sucessfully on the server you manage to connect to, to prevent any other group capturing that ID, and servers should monitor their fortune reponses to prevent duplicates on that server.

To receive bonus points, submit a brief report listing the ID's you found, and explaining the code you used to find it.

NB. Obviously there are collaborative solutions to this part of the assignment, however we will be monitoring for suspcious patterns in reporting, and checking the code.